BAB 2
TINJAUAN PUSTAKA

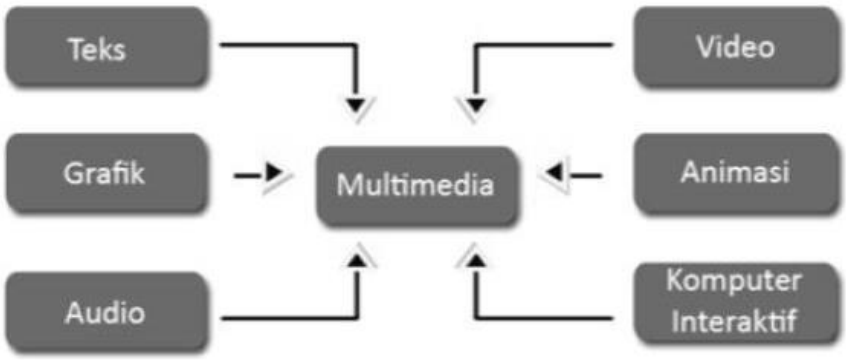
2.1 Tinjauan Pustaka

Bab ini menjelaskan tentang landasan teori dimana landasan teori ini menjelaskan dasar-dasar teori yang digunakan dalam penelitian yang dilakukan pada tugas akhir. Teori-teori yang digunakan meliputi multimedia, citra digital, *steganography*, *cryptography*, layanan *Multimedia Message Service* (MMS), aplikasi *mobile*, penjelasan umum SQLite, *Unified Modelling Language* (UML), teknologi *java*, *eclipse* dan *android*.

2.2 Multimedia

Multimedia berasal dari dua kata, yaitu multi dan media. Multi berarti banyak dan media bisa diartikan alat untuk menyampaikan atau membuat sesuatu, perantaraan, alat pengantar, suatu bentuk komunikasi seperti surat kabar, majalah atau televisi.

Apabila dikaitkan dengan pemrosesan komputer, media dianggap sebagai alat yang menampilkan teks, gambar, grafik, suara, musik dan sebagainya. *System* multimedia yang dimaksud disini adalah suatu teknologi yang menggabungkan berbagai sumber media seperti teks, grafik, suara, animasi, video dan sebagainya, yang disampaikan dan dikontrol oleh *system* komputer secara interaktif. Gabungan *system* tersebut bisa digambarkan pada gambar 2.1:



Gambar 2.1 Komponen Multimedia[2].

2.2.1 Keamanan Multimedia

Masalah keamanan merupakan salah satu aspek terpenting dari *system* multimedia. Masalah keamanan sering sekali kurang mendapatkan perhatian dari para perancang dan pengelola *system* informasi. Bahkan kadang berada di urutan kedua, ketiga atau di urutan terakhir dalam daftar hal-hal yang dianggap penting. Jika tidak mengganggu performa *system*, maka masalah keamanan sering kali tidak begitu dipedulikan, bahkan ditiadakan. Buku ini diharapkan dapat memberikan gambaran dan informasi menyeluruh tentang keamanan multimedia dan bisa membantu para perancang dan pengelola *system* informasi berbasis multimedia untuk mengamankan *system*nya.

Informasi menentukan hampir setiap elemen dari kehidupan manusia. Informasi juga sangat penting artinya bagai kehidupan ini. Tanpa adanya informasi, hampir semuanya tidak dapat dilakukan dengan baik. Contohnya, jika kita membeli tiket penerbangan dan membayarnya menggunakan kartu kredit, informasi mengenai *user* kartu disimpan dan dikumpulkan serta digunakan oleh pihak bank dan penerbangan. Demikian juga halnya saat kita membeli obat di apotek. Kita harus mendapat resep dari dokter dan memberikan resep tersebut kepada pelayan apotek. Hal ini juga merupakan salah satu informasi yang disampaikan oleh dokter kepada pihak apotek tentang obat apa yang kita butuhkan.

Hanya sedikit hal yang bisa dilakukan didunia modern tanpa melibatkan pengumpulan, penukaran, pembuatan atau pengaksesan informasi. Saat ini informasi sudah menjadi sebuah komoditi yang sangat penting. Bahkan ada yang menyatakan bahwa kita sudah berada dalam "*information-based society*". Kemampuan untuk mengakses dan meniadakan informasi secara cepat dan akurat menjadi sangat esensial bagi sebuah organisasi, baik organisasi komersial(perusahaan), perguruan tinggi, lembaga pemerintahan maupun pribadi. Hal tersebut semakin dimungkinkan yang juga disebut end user computing. Hal yang mendorong end user computing adalah meningkatnya pengetahuan manusia tentang komputer, banyaknya jasa informasi yang tersedia perangkat keras yang muara dan banyaknya perangkat lunak yang tersedia dipasaran.

Kemajuan *system* informasi memiliki banyak keuntungan didalam kehidupan manusia. Namun demikian, aspek negatifnya juga benak kehidupan manusia. Namun demikian, aspek negatifnya juga banyak terjadi, antara lain kejahatan komputer, yang meliputi pencurian, penipuan, pemerasan, kompetitif dan lain sebagainya. Jatuhnya informasi ke tangan pihak lain(misalnya pihak lawan bisnis) bisa menimbulkan kerugian bagi pemilik informasi. Sebagai contoh, banyak informasi dalam sebuah perusahaan yang hanya boleh diketahui oleh orang-orang tertentu didalam perusahaan tersebut, misalnya informasi tentang produk yang sedang dalam development, algoritma-Algoritma dan teknik-teknik yang digunakan untuk menghasilkan produk tersebut. Untuk itu, keamanan dari *system* yang digunakan harus terjamin dalam batas tertentu.

Pada zaman end user computing, hampir seluruh aspek kemasyarakatan menggunakan *system* informasi berbasis komputer. Apalagi informasi-informasi lebih mudah diperoleh dengan adanya jaringan komputer seperti LAN(*local Area Network*) dan internet, yang memungkinkan kita untuk menyediakan informasi secara tepat dan akurat[2].

Keamanan informasi di program mempunyai beberapa aspek yang harus dipenuhi, agar informasi dapat terjamin keaslian dan keamanannya. Aspek-aspek umum tersebut meliputi *confidentiality*, *integrity*, *authentication, availability* dan *non repudiation*[2].

1. Confidentiality

Confidentiality atau kerahasiaan yaitu keamanan informasi harus dapat menjamin kerahasiaan dari informasi. Salah satu cara yang dapat dilakukan yaitu dengan membatasi hanya orang yang mempunyai hak saja yang dapat membaca atau mengubah suatu informasi.

2. Integrity

Integrity berhubungan dengan keaslian informasi. *Integrity* yaitu menjamin keutuhan dan keaslian informasi yang dikirimkan, agar informasi tidak dirusak atau diubah oleh orang yang tidak berhak. Untuk menjaga integritas data, *system* harus memiliki kemampuan untuk mendeteksi manipulasi data oleh pihak-pihak

yang tidak berhak, antara lain *embedding*, penghapusan, dan pensubsitusian data lain kedalam data yang sebenarnya

3. *Authentication*

Authentication yaitu usaha atau metode untuk mengetahui keaslian dari informasi, dan memastikan bahwa informasi diterima oleh orang yang benar. Untuk menjaga otentikasi terhadap informasi, dapat digunakan *digital signature* untuk memastikan keaslian informasi.

4. Availability

Availability atau ketersediaan data yaitu keamanan informasi harus dapat menjamin bahwa data atau informasi harus tersedia ketika akan digunakan. Untuk menjamin ketersediaan data, *system* harus mempunyai cadangan data jika data tersebut hilang.

5. Non Repudiation

Non repudiation yaitu menjamin bahwa seorang *sender* informasi tidak dapat menyangkal keaslian dari informasi yang dikirimnya. Sehingga *receiver* informasi dapat memastikan bahwa informasi yang diterima merupakan informasi yang asli[5].

2.3 Penjelasan Umum Tentang Citra

Pada sub-bab ini akan dijelaskan secara umum mengenai teori dasar citra, macam citra, format citra digital dan pengolahan citra digital.

2.3.1 Citra Digital

Menurut Yusron (2011), citra digital merupakan fungsi intensitas cahaya f(x,y) pada bidang 2D, dimana harga x dan y merupakan koordinat spasial dan nilai fungsi tersebut pada setiap titik (x,y) merupakan tingkat kecemerlangan citra pada titik tersebut. Ada dua macam citra:

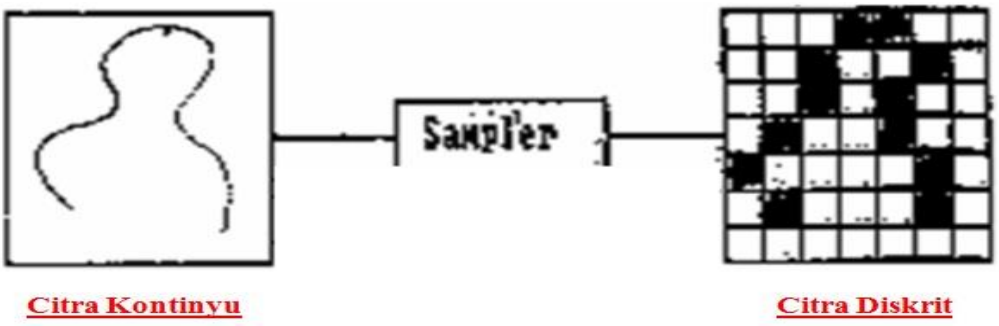
1. Citra Kontinu

- Dihasilkan dari *system* optik yang menerima sinyal analog
- Contoh : mata manusia, kamera analog

2. Citra Diskrit / Digital

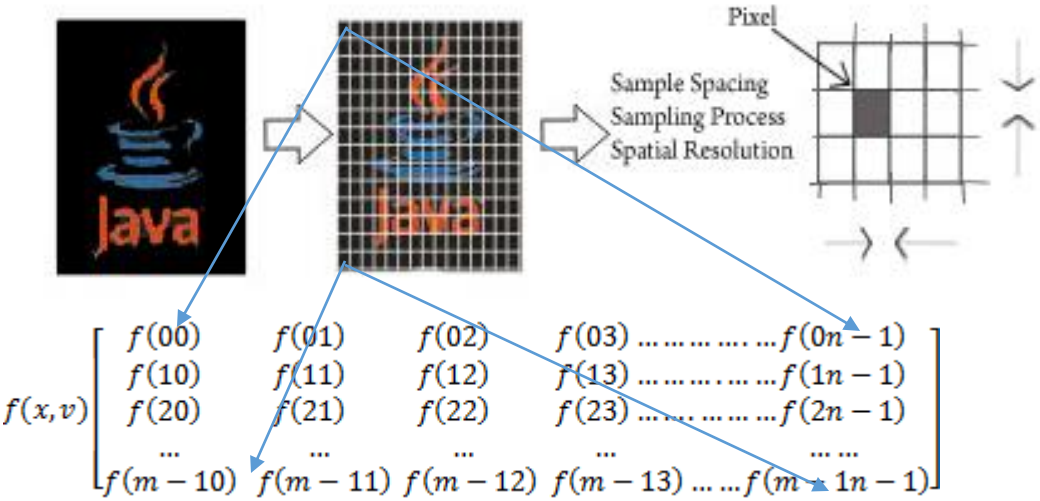
- Dihasilkan melalui proses digitalisasi terhadap citra kontinu

- Contoh : kamera digital, *scanner*[7].



Gambar 2.2 Citra Kontinu & Diskrit[7].

Berikut ilustrasi citra digital.



↓

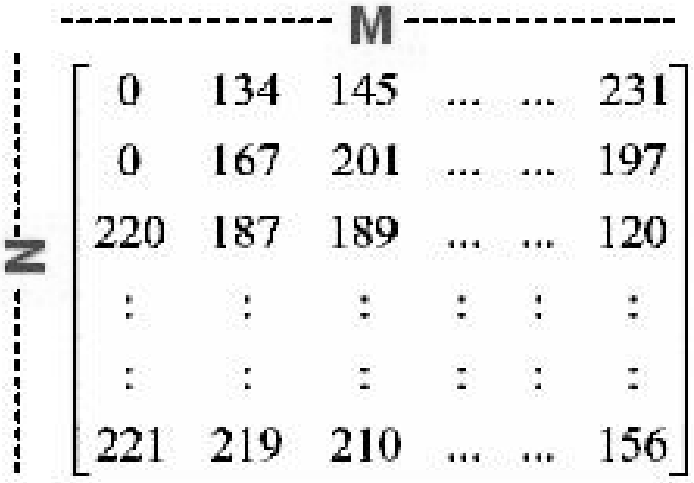
-Brightness Spacing

-Proses Kwantitasi

-Brightness Resolution

Gambar 2.3 Citra Digital[7].

Suatu citra berukuran N x M *pixel* dengan intensitas beragam pada tiap *pixel*nya, direpresentasikan secara numerik dengan matriks terdiri dari N baris dan M kolom.



Gambar 2.4 Matriks Pixel Pada Citra[7].

Berikut adalah ilustrasi *pixel*.

***Pixel* → 3** warna, yaitu(**RGB** - *RedGreenBlue*)

B (0-255)								**G (0-255)**								**R (0-255)**							
2^0	2^1	2^2	2^3	2^4	2^5	2^6	2^7	2^0	2^1	2^2	2^3	2^4	2^5	2^6	2^7	2^0	2^1	2^2	2^3	2^4	2^5	2^6	2^7

Sehingga total jumlah **RGB**=16,777,216 (256^3)



B (0-255)								**G (0-255)**								**R (0-255)**							
2^0	2^1	2^2	2^3	2^4	2^5	2^6	2^7	2^0	2^1	2^2	2^3	2^4	2^5	2^6	2^7	2^0	2^1	2^2	2^3	2^4	2^5	2^6	2^7







Gambar 2.5 Ilustrasi Pixel[7].

2.3.2 Format-format Citra Digital.

Macam-macam format gambar digital.

1. **PNG (Portable Network Graphics)**

Tipe *file PNG* merupakan solusi kompresi yang *powerfull* dengan warna yang lebih banyak (24 bit *RGB* + alfa). Berbeda dengan *JPEG* yang menggunakan teknik kompresi yang menghilangkan data, *file PNG* menggunakan kompresi yang tidak menghilangkan data (*lossles compression*). Kelebihan *file PNG* adalah adanya warna transparan dan *alfa*. Warna *alfa* memungkinkan sebuah gambar transparan, tetapi gambar tersebut masih dapat dilihat mata seperti samar-samar atau bening. *File PNG* dapat diatur jumlah warnanya 64 bit (*true color* + alfa) sampai index *color* 1 bit. Dengan jumlah warna yang sama, kompresi *file PNG* lebih baik daripada *GIF*, tetapi memiliki ukuran *file* yang lebih besar daripada *JPG*. Kekurangan tipe *PNG* adalah belum populer sehingga sebagian *browser* tidak mendukungnya.

File PNG cocok digunakan untuk:

1. Gambar yang memiliki warna banyak.
2. Gambar yang mau diedit ulang tanpa menurunkan kualitas[7].

2. **JPG/JPEG (Joint Photographic Experts Group)**

Tipe *file JPEG* sangat sering digunakan untuk web atau blog. *File JPEG* menggunakan teknik kompresi yang menyebabkan kualitas gambar turun (*lossy compression*). Setiap kali menyimpan ke tipe *JPEG* dari tipe lain, ukuran gambar biasanya mengecil, tetapi kualitasnya turun dan tidak dapat dikembalikan lagi. Ukuran *file JPEG* dapat turun menjadi sepersepuluhnya setelah dikonversi menjadi *JPEG*. Meskipun dengan penurunan kualitas gambar, pada gambar-gambar tertentu (misalnya pemandangan), penurunan kualitas gambar hampir tidak terlihat mata.

File JPEG cocok digunakan untuk:

1. Gambar yang memiliki banyak warna, misalnya foto wajah dan pemandangan.

2. Gambar yang memiliki gradien, misalnya perubahan warna yang perlahan-lahan dari merah ke biru.

File JPEG tidak cocok digunakan untuk:

1. Gambar yang hanya memiliki warna sedikit seperti kartun atau komik.
2. Gambar yang memerlukan ketegasan garis seperti logo[7].

2.4 Steganography

Kata *steganography* (*Steganography*) berasal dari kata Yunani. *Steganos* yang artinya ‘tersembunyi/terselubung’, dan *graphien,* ‘menulis’ sehingga kurang lebih artinya “menulis (tulisan) terselubung”[8].

Steganography adalah suatu teknik untuk menyembunyikan informasi yang bersifat pribadi dengan sesuatu yang hasilnya akan tampak seperti informasi normal lainnya. Media yang digunakan umumnya merupakan suatu media yang berbeda dengan media pembawa informasi rahasia, dimana disinilah fungsi dari teknik *steganography* yaitu sebagai teknik penyamaran menggunakan media lain yang berbeda sehingga informasi rahasia dalam media awal tidak terlihat secara jelas. *Steganography* juga berbeda dengan *cryptography* yaitu terletak pada hasil dari prosesnya. Hasil dari *cryptography* biasanya berupa data yang berbeda dari bentuk aslinya dan biasanya datanya seolah-olah berantakan namun dapat dikembalikan ke data semula. Sedangkan hasil dari keluaran *steganography* memiliki bentuk yang sama dengan data aslinya, tentu saja persepsi ini oleh indra manusia, tetapi tidak oleh komputer atau pengolah data digital lainnya. Namun secara umum *steganography* dan *cryptography* mempunyai tujuan yang sama yakni mengamankan data, bagaimana supaya data tidak dapat dibaca, dimengerti atau diketahui secara langsung. Media *cover* merupakan data *digital* yang akan ditempeli dengan data yang akan disembunyikan atau sering disebut dengan *stego* medium. Berbagai media yang dapat digunakan sebagai *cover* dari data atau informasi yang akan disembunyikan dengan berbagai teknik *steganography*.

Steganography dapat digunakan untuk berbagai macam alasan. Untuk tujuan legitimasi dapat digunakan pengamanan seperti citra dengan *watermarking*

dengan alasan untuk perlindungan hak cipta atau *copyright*. *Steganography* juga dapat digunakan sebagai cara untuk membuat pengganti suatu nilai hash satu arah (yaitu *user* mengambil suatu masukan panjang variabel dan membuat sebuah keluaran panjang statis dengan tipe string untuk melakukan verifikasi bahwa tidak ada perubahan yang dibuat pada variabel masukan yang asli). Selain itu juga, *steganography* dapat digunakan sebagai tag-notes untuk citra *online*. *Steganography* juga dapat digunakan untuk melakukan 3 perawatan atas kerahasiaan informasi yang berharga, untuk menjaga data tersebut dari kemungkinan sabotasi, pencuri, atau dari pihak yang tidak berwenang[8].

Berikut adalah kriteria *steganography* yang baik[7]:

1 *Imperceptible*

Keberadaan pesan tidak dapat dipersepsi oleh indra manusia. Jika pesan disisipkan ke dalam sebuah citra, citra yang telah disisipi pesan harus tidak dapat dibedakan dengan citra asli oleh mata. Begitu pula dengan suara, telinga haruslah mendapati perbedaan antara suara asli dan suara yang telah disisipi pesan.

2 *Fidelity*

Mutu media penampung tidak berubah banyak akibat *embedding*. Perubahan yang terjadi harus tidak dapat dipersepsi oleh indra manusia.

3 *Recovery*

Pesan yang disembunyikan harus dapat diungkap kembali. Tujuan *steganography* adalah menyembunyikan informasi, maka sewaktu-waktu informasi yang disembunyikan ini harus dapat diambil kembali untuk dapat digunakan lebih lanjut sesuai keperluan.

2.4.1 Sejarah Steganography

Kata *steganography* di Indonesia kurang populer dalam kehidupan sehari-hari. *Steganography* menjadi sangat populer setelah kasus pemboman gedung WTC di Amerika Serikat, teroris menyembunyikan pesan-pesan kegiatan teror dalam berbagai media yang dapat dijadikan penampung untuk menyembunyikan berkas seperti pada *image*, *audio*, dan *video*. Novel *Da Vinci Code* juga sempat mempopulerkan *steganography* dan *cryptography*.

Steganography berasal dari bahasa Yunani yaitu *steganos* yang artinya penyamaran atau persembunyian dan *graphein* yang artinya adalah tulisan Penyembunyian pesan ini dilakukan dengan tujuan agar hanya pihak *receiver* yang sah saja yang mengetahui bahwa terdapat pesan yang disembunyikan dalam media penampung[13].

Berbeda dengan *cryptography*, dimana pesan diubah dan diacak menjadi bentuk lain yang tidak bermakna. Pada *steganography*, pesan yang disembunyikan tetap dipertahankan hanya dalam penyampaiannya disembunyikan pada suatu media penampung. Pesan yang disampaikan secara *cryptography* menjadi mencurigakan karena berupa kata-kata yang tidak bermakna, sedangkan pesan dalam *steganography* terlihat seperti pesan biasa sehingga kecil kemungkinan untuk dicurigai.

Tehnik penyembunyian pesan ada bermacam-macam sejak jaman kuno. Menurut sejarah, pernah dilakukan oleh orang Yunani yang bernama Demeratus pada tahun 400 sebelum masehi dengan menulis pesan pada panel kayu yang kemudian disembunyikan dengan melapisi lilin sebagai penutupnya yang dikenal dengan istilah *wax tablets*[13].

Pada abad ke 5 sebelum masehi, Histaiacus mengirimkan pesan kepada Aristagoras Miletus untuk memberontak terhadap raja Persia. Pesan disampaikan dengan cara mencukur kepala pembawa pesan dan mental kepalanya dengan pesan tersebut. Setelah pesan dituliskan, pembawa pesan harus menunggu hingga rambutnya tumbuh kembali sebelum dapat mengirimkan pesan kepada pihak *receiver*. Pihak *receiver* kemudian akan mencukur rambut pembawa pesan tersebut untuk melihat pesan yang tersembunyi. Bangsa Romawi mengenal *steganography* dengan menggunakan tinta tak-tampak *(invisible ink)* untuk menuliskan pesan. Tinta tersebut dibuat dari campuran sari buah, susu, dan cuka. Jika tinta digunakan untuk menulis maka tulisannya tidak tampak. Tulisan di atas kertas dapat dibaca dengan cara memanaskan kertas tersebut[13].

2.4.2 *Haar Wavelet* transformasi

Dalam transformasi *Wavelet Haar*, terdapat dua proses yang harus dilakukan yaitu transformasi *forward* (dekomposisi) dan transformasi *Inversi* (rekontruksi). Transformasi *Forward* berguna untuk memecah gambar. Sedangkan transformasi *inversi* adalah kebalikannya, yaitu membentuk kembali pecahan-pecahan gambar dari proses *forward* menjadi sebuah citra seperti semula (proses rekonstruksi)[11].

2.4.3 Transformasi *Forward* (Dekomposisi)

Tiap langkah dalam transformasi *Haar* memperhitungkan kumpulan koefisien-koefisien *Wavelet* dan kumpulan rata-rata. Jika suatu kumpulan data $S_0,S_1«,S_{N-1}$ berisi unsur-unsur N, akan terdapat N/2 rata-rata dan N/2 nilai-nilai koefisien. Rata-rata disimpan dalam setengah lebih rendah dari kesatuan unsur N dan koefisien-koefisien disimpan dalam setengah diatas. Rata-rata menjadi input untuk langkah selanjutnya dalam penghitungan *Wavelet*, dimana untuk iterasi $_i+1,N_i+1=N_i/2$. Iterasi-iterasi berlanjut sampai suatu rata-rata tunggal dan koefisien tunggal dihitung.Ini mengganti sekumpulan data asal dari unsur-unsur N dengan rata-rata yang telah didapat, yang diikuti dengan sekumpulan koefisien-koefisien yang ukurannya adalah peningkatan pangkat dua(misalnya,$2^0,2^1,2^2$,«, N/2) (I. Dabechies 1992)[11].

Persamaan-persamaan *Haar* untuk menghitung suatu rata-rata (a_i) dan koefisien-koefisien *Wavelet* (c_i) dari suatu unsur ganjil dan genap dalam sekumpulan data ditunjukkan di bawah :

$$A_i = \frac{S_i + S_{i+1}}{2} \quad \text{.....(2.1)}$$

$$A_i = \frac{S_i - S_{i+1}}{2} \quad \text{.....(2.2)}$$

Dalam terminologi *Wavelet*, rata-rata *Haar* dihitung dengan fungsi penskalaan. Koefisien dihitung dengan fungsi *Wavelet*. Input data pada transformasi *Haar* dapat secara sempurna dibangun kembali dengan menggunakan persamaan-persamaan berikut :

$S_i = a_i + c_i$ .....(2.3)

$S_i = a_i - c_i$ .....(2.4)

Dalam pandangan aljabar linear transformasi forward *Haar*, rata-rata pertama dihitung dengan produk sinyal linear $[s_0,s_1,...s_{N-1}]$ dan vektor, dari ukuran yang sama [0,5,-0,5, 0, 0,...0]. Rata-rata dan koefisien selanjutnya dihitung dengan merubah penskalaan dan vektor-vektor *Wavelet* dengan dua dan menghitung produk-produk sebelah dalam. Pada penskalaan literatur dan nilai-nilai *Wavelet* terkadang ditunjukkan masing-masing dengan h_1 dan g_1. Koefisien-koefisien fungsi penskalaan:

$h_0=0,5$

$h_1=0,5$

Koefisien-koefisien fungsi *Wavelet* :

$g_0=0,5$

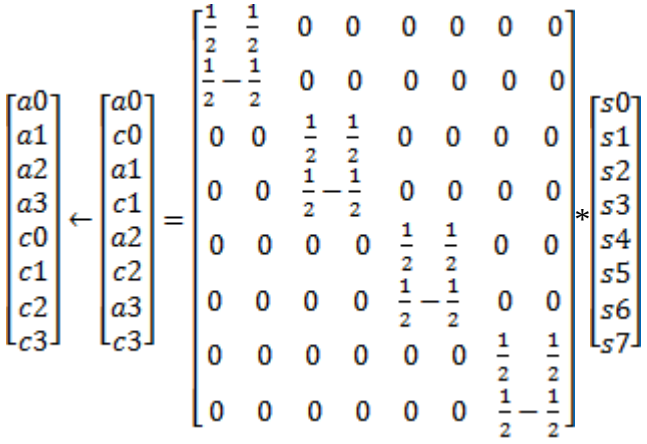
$g_1= -0,5$

Penskalaan dan nilai-nilai *Wavelet* untuk perubahan *Haar* ditunjukkan di bawah ini dalam bentuk matriks :

$$\begin{bmatrix} h_0 & h_1 & 0 & 0 & \cdots \\ g_0 & g_1 & 0 & 0 & \cdots \\ 0 & 0 & h_0 & h_1 & \cdots \\ 0 & 0 & g_0 & g_1 & \cdots \\ \vdots & \vdots & \vdots & \vdots & \ddots \end{bmatrix}$$

Gambar 2.2.6 Matriks Transformasi Haar[11].

Langkah pertama dari transformasi forward *Haar* delapan sinyal unsur diperlihatkan di bawah. Disini sinyal dikalikan dengan matriks transformasi *forward* :



Gambar 2.2.7 Matriks Transformasi *Forward* Haar[11].

Tanda panah menunjukkan operasi pembagian yang mengatur kembali hasil sehingga nilai-nilai rata-rata berada dalam setengah pertama vektor dan koefisien-koefisien berada dalam setengah kedua vektor. Langkah selanjutnya mengalikan nilai-nilai a_i dengan matriks perubahan 4x4, yang menghasilkan dua rata-rata baru dan dua koefisien-koefisien baru yang akan menggantikan rata-rata dalam langkah pertama. Langkah terakhir mengalikan rata-rata baru inisemua dengan matriks 2x2 yang menghasilkan rata-rata akhir dan koefisien akhir[11].

Citra asli V dengan M x N *pixel* didekomposisi menjadi empat subband LL^1, LH^1, HL^1, dan HH^1 dengan menggunakan transformasi *Wavelet Haar*. Komponen-komponen tersebut secara matematis untuk transformasi *Wavelet* dengan *filter Haar* dihasilkan dengan menggunakan persamaan sebagai berikut :

$ll_i(x,y) = \frac{1}{4}\sum_{i=0}^{1}\sum_{j=0}^{1} v(2x+i, 2y+j)$

$ll_i(x,y) = \frac{1}{4}\sum_{i=0}^{1} v(2x+i, 2y) - \frac{1}{4}\sum_{i=0}^{1} v(2x+i, 2y+1)$

$hh_i(x,y) = \frac{1}{4}\sum_{j=0}^{1} v(2x, 2y+j) - \frac{1}{4}\sum_{j=0}^{1} v(2x+1, 2y+j)$

$hh_1(x,y) = \frac{1}{4}$

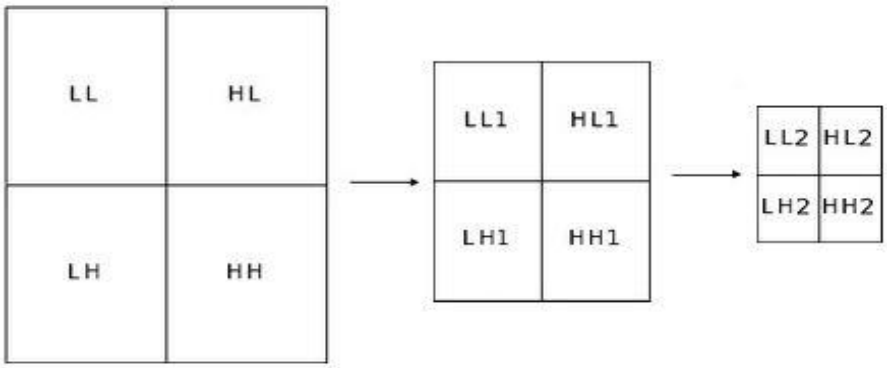
$\{v(2x,2y) + (v2x+1,2y+1) - v(2x+1,2y) - v(2x,2y+1)\}$.....(2.5)

Dengan syarat : $0 \leq x \leq \frac{M}{2}, 0 \leq y \leq \frac{N}{2}$.....(2.6)

Dimana v(x,y) merupakan nilai *pixel* pada koordinat(x,y) pada citra V. Sedangkan $ll_1(x, y)$, $lh_1(x,y)$, $hl_1(x,y)$, dan $hh_1(x,y)$ secara berturut-turut adalah komponen pada koordinat (x,y) dari LL_1,LH_1,HL_1, dan HH_1.LLmerupakan

setengah resolusi dari citra asli. LH merupakan subband detail horizontal,HL merupakan subband detail vertikal, dan HH merupakan subband dari detail diagonal. LL_1selanjutnya didekomposisi menjadi empat subband LL_2, HL_2, LH_2, dan HH_2. Operasi ini dapat diulang sampai dengan LL sama dengan1x1[11].

Berikut adalah gambar dan pembagian subband dari proses forwad *Haar* :



Gambar 2.8 Proses Transformasi *Forward* Haar[11]

Keterangan:

L = Lowpass H = Highpass

2.4.4 Transformasi *Inverse*(Rekontruksi)

Seperti pada transformasi forward *Haar*, satu langkah dalam transformasi *invers*i *Haar* dapat digambarkan dalam hubungan-hubungan aljabar linear. Operasi matriks untuk membalikkan langkah pertama transformasi *Haar* untuk delapan sinyal unsur ditunjukkan dibawah :

$$\begin{bmatrix} s0 \\ s1 \\ s2 \\ s3 \\ s4 \\ s5 \\ s6 \\ s7 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 \end{bmatrix} * \begin{bmatrix} a0 \\ c0 \\ a1 \\ c1 \\ a2 \\ c2 \\ a3 \\ c3 \end{bmatrix} \leftarrow \begin{bmatrix} a0 \\ a1 \\ a2 \\ a3 \\ c0 \\ c1 \\ c2 \\ c3 \end{bmatrix}$$

Gambar 2.9 Matriks Transformasi *Inverse* Haar[11].

Proses pengembalian dekomposisi *Haar* menjadi sebuah citra kembali (rekonstruksi), secara singkat bisa dijabarkan sebagai berikut:

1. Mengembalikan LL2, HL2, LH2, dan HH2 menjadi LL1, dengan cara mengambil satu *pixel* di LL2, HL2, LH2, dan HH2, dengan koordinat yang sama, begitu seterusnyasampai koordinat terakhir.
2. Mengembalikan LL1, HL1, LH1, dan HH1 menjadi citra kembali dengan cara yang sama seperti pada proses.[11]

Haar wavelet transformasi ditemukan pada tahun 1909. Dimana *haar wavelet* juga dikenal sebagai *mother wavelet.* (atau sebuah *wavelet* induk yang telah digunakan sejak pertama kali secara intensif) yang didefinisikan sebagai berikut[8]:

$$\psi(x) = \begin{cases} 1{,}0 \ \leq x \leq \frac{1}{2} \\ -\frac{1{,}1}{2} \leq x < 1 \\ 0, x \ yang \ lain \end{cases} \quad \text{.....(2.7)}$$

dan

$$\phi(x) = \begin{cases} 1{,}0 \ \leq x \leq 1 \\ 0, yanglain \end{cases} \quad \text{.....(2.8)}$$

Transformasi *Haar Wavelet* merupakan transformasi *wavelet* paling sederhana. Dalam transformasi *Haar Wavelet* pada suatu *image* dilakukan dengan menggunakan penapis lolos rendah (*Low Pass Filter* / LPF) dan penapis lolos tinggi (*High Pass Filter* / HPF) sehingga diperoleh koefisisen *wavelet* [Ogden, R.T., 1997][8].

Algoritma:

1. Input: *image* ternormalisasi
2. Untuk masing–masing dekomposisi horisontal dan vertikal, cari koefisien LPF dan HPF. Koefisien LPF adalah:

   $f_k^1 = 1/\sqrt{2}(f_{2k} + f_{2k-1})$ .....(2.9)

   Koefisien HPF adalah:

   $f_k^1 = 1/\sqrt{2}(f_{2k} - f_{2k-1})$ .....(2.10)
3. Lakukan secara berulang–ulang pada koefisien aproksimasi yang diperoleh sebelumnya hingga level yang diinginkan.

Dimana keuntungan 2D *Haar Wavelet* diantaranya adalah[8]:

1. Waktu komputasi cepat.
2. Sederhana.
3. Merupakan metode yang efisien.

Transformasi *wavelet* adalah sebuah konsep yang relatif baru. Ada dorongan terhadap *user wavelet* dalam pemrosesan sinyal dan analisis dari (atau selain) diskrit kosinus mengubah (DCT), yang digunakan dalam standar JPEG kompresi gambar. Baru-baru ini, banyak Algoritma telah mengusulkan untuk menggunakan *wavelet* untuk kompresi gambar. Teknik yang saat ini sedang digunakan dalam bekerja dengan gambar dapat umum untuk digunakan dengan transformasi *wavelet*. Ada banyak aplikasi untuk *wavelet*, dan *user wavelet* dalam pemrosesan sinyal tampaknya tak berujung. Mengubah *Haar* adalah salah satu paling sederhana transformasi *wavelet* matematika[10].

2.4.5 Transform *Haar* 1D

1 D *Haar* Transform bekerja pada set 4 *pixel*. Transformasi dari serangkaian *pixel*:

a	b	c	d

adalah

A = (e+f)/2	B = (e-f)/2	C = (a-b)/2	D = (c-d)/2

dimana e = (a + b) / 2 dan f = (c + d) / 2.

Transformasi *invers* adalah:

A + B + C	A + B - C	A - B + D	A + B - D

Generalisasi untuk dua dimensi, kita mengambil 1D *Haar* mengubah pertama sepanjang baris, kemudian sepanjang kolom gambar. Jika lebar atau tinggi tidak kelipatan 4, kemudian perbatasan diabaikan dalam mengubah.

Kami menemukan ini mengubah menjadi relatif tidak efektif dalam *Steganography*. Mengubah *Haar* disamaratakan dua dimensi memungkinkan lebih banyak data untuk disimpan dalam gambar biasa[10].

2.4.6 *Haar* 2Dnd mengubah tingkat

2D *Haar* Transform juga bekerja pada set 4 *pixel*, tetapi dianggap "2D" karena ada tambahan pengolahan di blok 2 x 2 setelah awal baris dan kolom transformasi selesai. Transformasi dari serangkaian *pixel*:

a	b	c	d

adalah

A = (a+b)/2	B = (c+d)/2	C = (a-b)/2	D = (c-d)/2

Sekali lagi, perbatasan untuk gambar dengan lebar atau tinggi yang tidak kelipatan empat diabaikan.

Seperti mengubah 1 D, transformasi adalah umum untuk dua dimensi, pertama sepanjang baris, maka sepanjang kolom. Selain itu, setelah perhitungan ini selesai, kita perlu bekerja pada sudut kiri atas (2 x 2 *pixel*) setiap 4 x 4 blok.

e	f

menjadi

E = (a+b)/2	F = (a-b)/2

Perhitungan sama dilakukan untuk baris kedua *pixel*, dan kemudian kembali untuk setiap kolom ini 2 x 2 blok.

Invers pertama dihitung untuk setiap grup 4 x 4 *pixel*. Untuk melakukan ini, kita mengambil *invers* untuk pertama dua kolom di blok 2 x 2, kemudian mengambil *invers* untuk dua baris pertama dalam 2 x 2 blok. *Invers* adalah:

E + F	E - F

, untuk E dan F yang ditunjukkan di atas.

Akhirnya, *invers* diambil untuk setiap grup 4 *pixel*; pertama oleh kolom, kemudian oleh baris, sebagai berikut

A + C	A - C	B + D	B - D

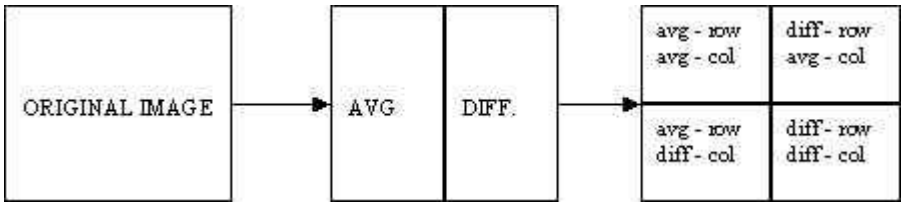
untuk A, B, C, dan D di atas.

Transformasi ini digunakan dalam berbasis *Tear Steganography*. Jumlah data yang dapat disembunyikan dalam gambar tergantung pada jumlah koefisien "rendah" yang dihasilkan oleh transformasi[10].

2.4.7 *Generalized Lth* didekomposisi *Haar Transform*

Untuk *watermarking*, mengubah *Haar* dua dimensi yang dibahas di atas umum untuk jumlah yang sewenang-wenang dekomposisi. Sedangkan transformasi dua dimensi bekerja pada sebuah 4 x 4 blok *pixel*, Lth tingkat dekomposisi bekerja pada 2n x 2m gambar. Setiap *pixel* yang aneh di perbatasan

diabaikan. Untuk setiap pasangan *pixel*, rata-rata disimpan pada paruh pertama dari baris atau kolom, dan perbedaan disimpan dalam paruh kedua baris atau kolom. Sekali lagi, kami mengambil mengubah dua arah - pertama sepanjang setiap baris, kemudian sepanjang setiap kolom. Setelah langkah ini, kita telah disimpan rata-rata di kuadran kiri atas gambar:



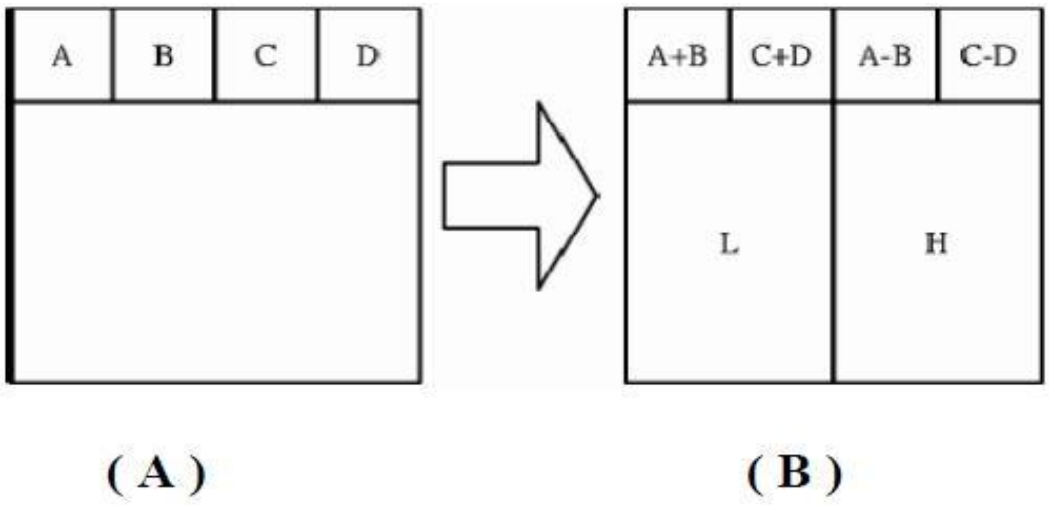
Gambar 2.10 *Generalized Lth* didekomposisi Haar *Transform*

Kita bisa melakukan dekomposisi sama di kuadran kiri atas hingga kali L, dimana L adalah log2(min (tinggi, lebar)). Asumsi di sini adalah bahwa kebanyakan *pixel* "penting" bisa disimpan di kuadran kiri atas dari setiap tingkat dekomposisi, dan "detail" didorong ke dalam kuadran tiga lainnya[10].

2.4.8 Metode Haar Wevelet

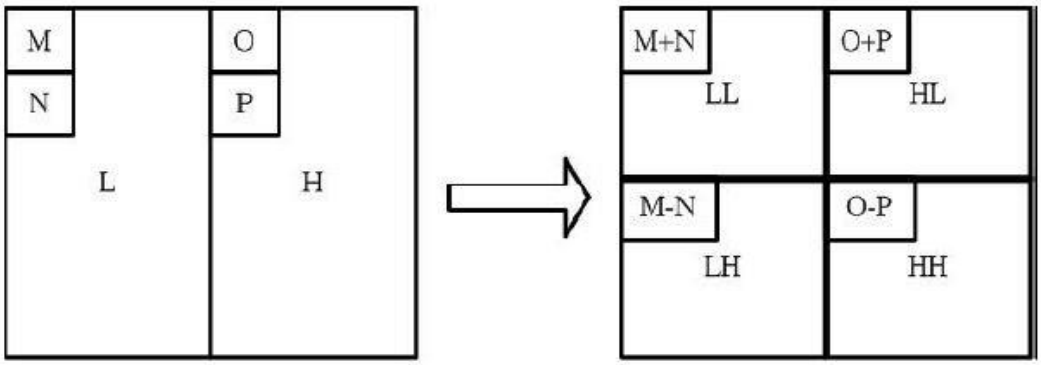
Pembangunan aplikasi ini menggunakan metode *steganography Haar Wevelet. Domain* frekuensi transformasi yang diterapkan dalam penelitian ini adalah *Haar Wavelet – DWT* (*Discrete Wavelet transform*), *Discrete Wavelet Transform* yang paling sederhana adalah *Haar* 2-dimensi-*DWT* dimana terdiri dari dua operasi yaitu: *operation* secara horisontal dan vertikal. Tahapan perhitungan pada 2-D *Haar-DWT* adalah sebagai berikut:

Langkah 1: Pada awalnya, *pixel* di *scan* dari kiri ke tepat di arah horisontal. Lalu, lakukan operasi penjumlahan dan pengurangan pada *pixel* sebelahnya. Tempatkan penjumlah di sebelah kiri dan pengurangan di sebelah kanan seperti yang diilustrasikan pada Gambar 2.11 Ulangi operasi ini sampai semua baris diproses. penjumlah pada *pixel*mewakili bagian frekuensi rendah (dinotasikan sebagai simbol L) sedangkan pengurangan pada *pixel*mewakili frekuensi tinggi sebagai bagian dari gambar asli (dinotasikan sebagai simbol H).



Gambar 2.11 *Operation Horizontal*, B. Penempatan Operasi L dan H

Langkah 2: kedua, *pixel* I di scan dari atas ke bawah dalam arah vertikal. Melakukan operasi penjumlahan dan operasi pengurangan di *pixel* tetangga dan kemudian menyimpan penjumlah di atas dan pengurangan di bagian bawah seperti yang diilustrasikan pada Gambar 2.12 . Ulangi operasi ini sampai semua kolom diproses. Akhirnya kita akan mendapatkan 4 sub-band dinotasikan sebagai LL, HL, LH, dan HH masing. Sub LL-band adalah frekuensi rendah bagian dan karenanya terlihat sangat mirip untuk gambar asli.



Gambar 2.12 *Operation* Secara Vertikal

Seluruh prosedur yang dijelaskan di atas adalah disebut orde pertama 2-D *Haar*-DWT. Orde pertama 2-D *Haar*-DWT diterapkan pada gambar "Java" diilustrasikan pada Gambar 2.13:



(A) **(B)**

Gambar 2.13 *Image Original* - Java, B. Hasil Dari Orde Pertama 2-D Haar-DWT.

Saat *DWT* pada *image* pembawa berlangsung, didapatkan 4 *subband*, yakni LL, LH, HL, dan HH. *Sub-band* LL mempresentasikan *image* aproksimasi yakni *image* yang sama dengan *image* aslinya tetapi berukuran setengah dari *image* asli. *Sub-band* ini memiliki energi yang besar tetapi faktor *hidden*-nya kurang. Sebaliknya *sub-band* HH memiliki energi yang kecil namun memiliki faktor *hidden*-nya yang baik. Sedangkan kedua *sub-band* lainnya yaitu LH dan HL memiliki energi yang sedang dan faktor *hidden* yang tidak terlalu baik pula. Dalam proses ini dilakukan pemilihan *sub-band* yang akan digunakan sebagai tempat penyembunyian informasi.

2.4.9 Algoritma *Haar Wavelet*

Transformasi *Haar* kembali banyak koefisien yang 0, atau menutup ke 0. Setelah mengambil *Haar* mengubah beberapa gambar, ide adalah untuk menyembunyikan bit di koefisien yang di bawah beberapa nilai ambang batas, kemudian simpan *invers Haar* diubah data. Secara teoritis, karena kita bersembunyi bit di koefisien tidak signifikan, ini harus sangat memodifikasi gambar[10].

Encoding:

1. Mengambil (2D *Haar*) Tear mengubah gambar.
2. Mencari koefisien di bawah nilai ambang batas.
3. Mengganti bit ini dengan bit data tersembunyi.
4. Mengambil mengubah *invers*.

5. Menyimpannya sebagai gambar biasa, format standar, seperti .pgm.

Decoding:

1. Mengambil transformasi *wavelet* gambar.
2. Mencari nilai koefisien yang di bawah ambang batas nilai.
3. Meng*extract* bit data dari koefisien ini.
4. Menggabungkan bit data di*extract*si ke pesan aktual.
5. Keluaran pesan atau data.

2.5 Cryptography

Cryptography adalah ilmu yang mempelajari teknik-teknik matematika yang berhubungan dengan aspek keamanan informasi seperti kerahasiaan, integritas data, otentikasi, dan autentifikasi data. *Cryptography* bukan satu-satunya cara untuk menyediakan keamanan informasi, melainkan satu set teknik yang dapat digunakan untuk mengamankan informasi[2].

Secara umum, *cryptography* terdiri dari dua buah bagian utama yaitu bagian *encrypt* dan bagian *decrypt*. *Encrypt* adalah proses transformasi informasi menjadi bentuk lain sehingga isi pesan yang sebenarnya tidak dapat dipahami, hal ini dimaksudkan agar informasi tetap terlindung dari pihak yang tidak berhak menerima. Sedangkan *decrypt* adalah proses kebalikan *encrypt*, yaitu transformasi data ter*encrypt* ke data bentuk semula. Proses transformasi dari *plaintext* menjadi *cipherteks* akan dikontrol oleh kunci. Peran kunci sangatlah penting, kunci bersama-sama dengan Algoritma matematisnya akan memproses *plaintext* menjadi *cipherteks* dan sebaliknya.

Cryptography tidak memenuhi semua aspek dari keamanan informasi. *Cryptography* hanya memenuhi empat aspek dalam keamanan informasi yang merupakan tujuan dari *cryptography*. Keempat aspek tersebut yaitu kerahasiaan (*confidentiality*), integritas data (*integrity*), otentikasi data (*authentication*), dan *non-repudiation*[5].

1. Kerahasiaan

Kerahasiaan adalah layanan yang digunakan untuk menjaga isi dari informasi dari siapapun kecuali yang memiliki otoritas atau kunci rahasia untuk membuka

informasi yang telah disandi. *Cryptography* memenuhi aspek kerahasiaan karena informasi tidak dapat secara langsung diketahui.

2. Integritas data

Integritas data adalah layanan yang berhubungan dengan penjagaan dari perubahan data secara tidak sah. Untuk menjaga integritas data, *system* harus memiliki kemampuan untuk mendeteksi manipulasi data oleh pihak-pihak yang tidak berhak, antara lain *embedding*, penghapusan, dan pensubsitusian data lain kedalam data yang sebenarnya.

3. Otentikasi

Otentikasi adalah layanan yang berhubungan dengan identifikasi/ pengenalan, baik secara kesatuan *system* maupun informasi itu sendiri. Dua pihak yang saling berkomunikasi harus saling memperkenalkan diri. Informasi yang dikirimkan melalui jaringan harus diotentikasi keaslian, isi datanya, waktu *sender*, dan lain-lain. Untuk alasan ini aspek *cryptography* biasanya dibagi menjadi dua kelas utama yaitu otentikasi entitas dan otentikasi data asal.

4. Non-repudiation

Non-repudiation adalah layanan yang mencegah terjadinya penyangkalan terhadap *sender*/terciptanya suatu informasi oleh yang mengirimkan/ membuat. Sebagai contoh, satu entitas dapat mengizinkan pembelian properti oleh entitas lain dan kemudian berusaha menyangkal otorisasi tersebut diberikan. Sebuah prosedur yang melibatkan pihak ketiga yang terpercaya diperlukan untuk menyelesaikan sengketa tersebut.

2.5.1 Algoritma Cryptography

Algoritma *cryptography RSA* adalah Algoritma untuk keamanan data yang ditemukan oleh Rivest, Shamir dan Adleman (*RSA*) pada tahun 1977-1978.

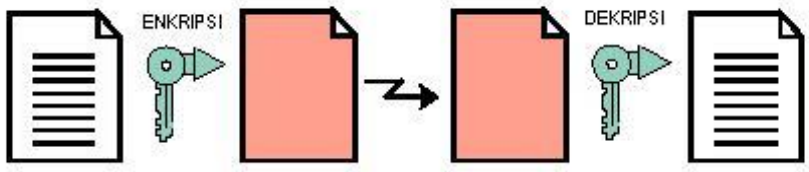
Menurut M. Zaki Riyanto dan Ardhi Ardhian (*Cryptography Public key*[14]) : Keamanan sandi *RSA* terletak pada sulitnya memfaktorkan bilangan yang besar. Selain sulitnya memfaktorkan bilangan yang besar pada pembentukan kunci pada *RSA*, *RSA* menawarkan kemampuan untuk mengendalikan panjang

kunci yang digunakan oleh *user*nya sehingga dapat menyesuaikan dengan kebutuhan *user*.

Algoritma *cryptography* yang handal adalah Algoritma *cryptography* yang kekuatannya terletak pada kunci, bukan pada kerahasiaan Algoritma itu sendiri. Berdasarkan jenis kuncinya, Algoritma *cryptography* dibagi menjadi dua jenis yaitu Algoritma *simetris* dan Algoritma *asimetris*.

2.5.1.1 Algoritma *Simetris*

Algoritma *simetris* (*symmetric algorithm*) adalah suatu Algoritma dimana kunci *encrypt* yang digunakan sama dengan kunci *decrypt* sehingga Algoritma ini disebut juga sebagai *single-key algorithm*[5]. Ilustrasi *user* Algoritma *cryptography* dengan kunci *simetris* dapat terlihat pada gambar 2.14.

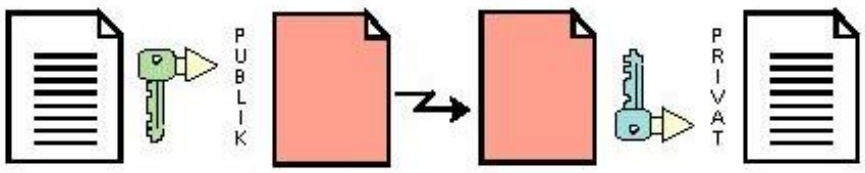


Gambar 2.14 Ilustrasi *Cryptography* Dengan Kunci Simetris

Algoritma *simetris* banyak digunakan karena lebih cepat dan lebih simpel, namun *user* kunci simetris juga mempunyai kekurangan, karena jika kunci dapat diketahui, maka informasi pun dapat diketahui. Beberapa Algoritma *cryptography* yang termasuk pada Algoritma *simetris* yaitu DES, AES, *Rijndael*, *Blowfish*, dan IDEA.

2.5.1.2 Algoritma *Asimetris*

Algoritma *asimetris* (*asymmetric algorithm*) adalah suatu Algoritma dimana kunci *encrypt* yang digunakan tidak sama dengan kunci *decrypt*[5]. Algoritma ini menggunakan dua kunci yakni *public key* (*public key*) dan kunci privat (*private key*). *Public key* disebarkan secara umum sedangkan kunci privat disimpan secara rahasia oleh *user*. Ilustrasi *user* Algoritma *cryptography* dengan kunci *asimetris* dapat terlihat pada gambar 2.15.



Gambar 2.15 Ilustrasi *Cryptography* Dengan Kunci Asimetris.

Algoritma *asimetris* mempunyai keamanan yang lebih baik, karena jika *public key* diketahui, informasi belum tentu dapat diketahui karena *private key* kemungkinan berbeda. Namun akan menjadi sulit dan lama ketika implementasinya. Beberapa Algoritma *cryptography* yang termasuk pada Algoritma *asimetris* yaitu Diffie – Hellman, *RSA*, ElGamal, dan DSA.

2.5.2 Algoritma *RSA*

Menurut (Deffie & Hellman, 1976) dari sekian banyak metode *cryptography asimetris* yang pernah dibuat, Algoritma yang paling popular adalah Algoritma *RSA*. Algoritma *RSA* diciptakan oleh 3 orang peneliti dari MIT (Masschussets Institute of Technology) pada tahun 1976, yaitu Ron (R)ivest, Adi (S)hamir dan Leonard (A)dleman. Keamanan Algoritma *RSA* terletak pada sulitnya menfaktorkan bilangan yang besar menjadi faktor – faktor prima. Penfaktoran dilakukan untuk memperoleh *private key*. Selama penfaktoran bilangan besar menjadi bilangan prima belum tentu menemukan Algoritma yang benar, maka selama itu pula keamanan Algoritma *RSA* terjamin.

Algoritma *RSA* memiliki besaran seperti berikut :

1. p dan q bilangan prima (rahasia)
2. n = p.q (tidak rahasia)
3. Φ(n) = (p-1)(q-1) (rahasia)
4. e = kunci *encrypt* (tidak rahasia)
5. d = kunci *decrypt* (rahasia)
6. m = *plaintext* (rahasia)
7. c = chipertext (tidak rahasia)

2.5.2.1 Algoritma Pembangkit Kunci

Berikut langkah-langkah membangkitkan kunci Algoritma *RSA* :

1. Menentukan 2 buah bilangan prima acak dan dirahasiakan, dimisalkan kedua bilangan prima tersebut diwakilkan oleh variabel *p* dan *q*. Dengan syarat nilai *p* ≠ nilai *q*.

   Semakin besar bilangan prima *p* dan *q*, maka kemungkinan untuk dilakukannya faktorisasi dengan metode *brute force* guna mencari nilai *p* dan nilai *q* nantinya akan semakin kecil.
2. Menentukan (*RSA*) modulus yang berfungsi sebagai salah satu pasangan *public key* maupun kunci rahasia. (*RSA*) modulus di wakilkan dengan variabel *n*. Nilai *n* didapatkan dari perkalian dua buah bilangan prima yang ditentukan pada langkah sebelumnya, yang diwakilkan oleh variabel p dan variabel *q*. Secara *system*atis, formula mencari nilai (*RSA*) modulus adalah sebagai berikut :

   $$n = p * q$$
3. Langkah selanjutnya adalah menentukan (*RSA*) *enciphering exponent* yang nantinya digunakan sebagai pasangan *public key* bersama (*RSA*) modulus. (*RSA*) *enciphering exponent* diwakilkan oleh variabel *e*. Formula untuk menentukan nilai *e* sebagai berikut :

   $$gcd(e, m) = 1$$

   dengan syarat 1 < *e* < *m* dan *e* adalah bilangan prima. *gcd* adalah akronim dari *greatest common divisor* atau faktor persekutuan terbesar (FPB). Bila gcd atau FPB dari dua buah bilangan prima terhadap nilai *m*, sehingga nilai *e* bisa digunakan untuk menjadi pasangan *public key* bersama (*RSA*) modulus.

   Formula penentuan nilai (*RSA*) *enciphering exponent* yang disebutkan sebelumnya hanya bisa dijalankan bila nilai *m* telah ditentukan.

   Formula untuk mencari nilai *m* sendiri adalah : bernilai = 1, misalnya *gcd(e,m)= 1*, artinya nilai *e* merupakan bilangan yang relatif prima terhadap nilai *m*, sehingga nilai *e* bisa digunakan untuk menjadi pasangan *public key* bersama (*RSA*) modulus.

Formula penentuan nilai (*RSA*) *enciphering exponent* yan disebutkan sebelumnya hanya bisa dijalankan bila nilai m telah ditentukan.

Formula untuk mencari nilai m sendiri adalah :

$$m = (p-1) * (q-1)$$

4. Langkah selanjutnya adalah menentukan nilai (*RSA*) *deciphering exponent* yang diwakilkan oleh variabel *(d),* (*RSA*) *deciphering exponent* berfungsi sebagai pasangan (*RSA*) modulus dalam pembentukan kunci rahasia *(n,d).* Formula mencari nilai *d* adalah sebagai berikut :

$$d = e^{-1} \, mod \, m$$

Nilai dari vaiabel *e* dan variabel *m* adalah nilai yang didapat dari langkah sebelumnya dalam pembentukan (*RSA*) *enciphering exponent.*

Dengan syarat *1 < d < m.*

Dalam praktiknya, rumus baku untuk mencari nilai *d* diatas dijabarkan lebih lanjut sebagai berikut :

$d = e^{-1} \, mod \, m$

$e * d = 1 * (mod \, m)$

$e * d = 1 + (k * m)$

$$e * d = 1 + (k * m)$$

Variabel *k* adalah bilangan *integer* dengan nilai sembarang yang diharapkan menghasilkan *d* yang bernilai *integer* juga karena nilai d yang akan diambil jika d = *integer*.

5. Nilai variabel *d, p, q* dan *m* adalah nilai yang harus dirahasiakan. Nilai variabel *n, e* adalah nilai yang boleh dipublikasikan.

   Pasangan *(n,e)* merupakan *public key* yang boleh dipublikasikan Pasangan *(n,d)* merupakan kunci rahasia yang harus dirahasiakan.

2.5.2.2 Algoritma *Encrypt*

Untuk melakukan *encrypt* menggunakan *public key* (n,e) digunakan formula *encrypt* sebagai berikut :

$$C = M^e \bmod n$$

Dimana :

c = *Ciphertext* / text yang telah di *encrypt*

m = *Plaintext* / Pesan asli

e = *Enciphering Exponent*

n = (*RSA*) modulus

2.5.2.3 Algoritma *Decrypt*

Untuk melakukan *decrypt* suatu *chiphertext* yang telah di *encrypt* sebelumnya, digunakan pasangan kunci rahasia (n,d), formula *decrypt* yang digunakan sebagai berikut :

$$M = C^d \bmod n$$

Dimana :

C = *Ciphertext* / text yang telah di *encrypt*

M = *Plaintext* / Pesan asli

e = *Enciphering Exponent*

n = (*RSA*) modulus

2.6 Penilaian Secara Obyektif

Kriteria penilaian secara obyektif berdasarkan pengukuran *error* pada citra yang diuji. Parameter-parameter yang digunakan sebagai kriteria penilaian obyektif adalah sebagai berikut[6] :

1. *Mean Square Error (MSE)*

   Rumus MSE dapat dituliskan sebagai berikut :

$$\text{MSE} = \frac{1}{M.N} \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} \left(f(x,y) - g(x,y) \right)^2 \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad (2.11)$$

2. *Peak Signal to Noise Ratio (PSNR)*

   Peak Signal to Noise Ratio (PSNR) adalah perbandingan antara nilai maksimum dari sinyal yang diukur dengan besarnya derau yang berpengaruh pada sinyal tersebut. PSNR merupakan parameter standar untuk menilai kualitas suatu citra secara obyektif dengan membandingkan *noise* terhadap sinyal puncak.

   Nilai PSNR yang baik biasanya berkisar antara 20dB sampai dengan 40dB. Pada umumnya disajikan dengan angka desimal yaitu dua angka dibelakang koma. Nilai nyata (*actual value*) tidak sepenuhnya berarti, tetapi perbandingan dua nilai untuk citra rekonstruksi yang berbeda memberikan satu nilai mutu.

   PSNR biasanya diukur dalam satuan desibel. Rumus PSNR adalah sebagai berikut[6] :

$$\text{PSNR} = 20.\log 10 \left(\frac{255}{\sqrt{MSE}} \right) \quad \text{..... (2.12)}$$

2.7 Layanan Pesan Multimedia (MMS)

Layanan pesan multimedia (MMS/*Multimedia Messaging Service*) merupakan puncak dari evolusi layanan pesan singkat (SMS/*Short Messaging Service*) dan EMS (*Enhanced Messaging Service*). MMS menawarkan perkembangan aplikasi secara menyeluruh sehingga pesan yang dikirim lebih kreatif dan menghibur. Pesan MMS dapat berupa teks, grafik/citra, data, animasi, audio, dan video.

Tidak seperti SMS yang menggunakan sinyal *link* dengan kapasitas minim, MMS menggunakan *main data channel* seperti GPRS (*General Packet Radio Service*). Pesan disimpan sementara pada MMSC jika pemakai tidak segera men-*download*. Pada MMS terdapat fitur *user profile* dan konversi data yang bergantung pada spesifikasi alat komunikasi (telepon selular) dan pemakai. *User profile* penting untuk mengetahui waktu ketika pesan di-*download*, karena pesan MMS terlalu besar sehingga tidak bisa segera sampai ke *receiver*. Kapasitas rata-

rata SMS sekitar 140 byte. Sedangkan kapasitas rata-rata pada MMS sekitar 30.000 byte bahkan bisa mencapai 100.000 byte. Pesan yang dikirim tergantung kemampuan jaringan selular dan telepon selular[6].

2.8 Aplikasi Mobile

Menurut Romdoni (2010), aplikasi adalah program yang digunakan untuk melakukan sesuatu pada *system* komputer. *Mobile* dapat diartikan sebagai perpindahan yang mudah dilakukan dari suatu tempat ke tempat lain, misalnya telepon genggam, yang dapat digunakan dengan berpindah-pindah tempat dengan mudah dari suatu tempat ke tempat lain tanpa pemutusan atau terputusnya komunikasi.

System aplikasi *mobile* adalah aplikasi yang dapat digunakan *user* dengan berpindah-pindah tempat dengan mudah dari suatu tempat ke tempat lain tanpa pemutusan atau terputusnya komunikasi.

Sedangkan akses informasi dari aplikasi *mobile* biasanya hanya berupa teks sederhana, kalaupun berupa gambar, maka gambar dengan ukuran yang tidak terlalu besar. Perangkat *mobile* yang digunakan masa kini seperti ponsel cerdas, tablet PC, PDA, dll. Beberapa karakteristik perangkat *mobile*:

1 Ukuran yang kecil

  Perangkat *mobile* memiliki ukuran yang kecil.Konsumen menginginkan perangkat yang terkecil untuk kenyamanan dan mobilitas mereka.

2 *Memory* yang terbatas

  Perangkat *mobile* juga memiliki *memory* yang kecil, yaitu primary (RAM) dan secondary (disk).

3 Daya proses yang terbatas

  System mobile tidaklah dibandingkan desktop.

4 Mengkonsumsi daya yang rendah

  Perangkat *mobile* menghabiskan sedikit daya dibandingkan dengan mesin desktop.

5 Kuat dan dapat diandalkan

  Karena perangkat *mobile* selalu dibawa kemana saja, mereka harus cukup kuat

untuk menghadapi benturan-benturan, gerakan, dan sesekali tetesan-tetesan air.

6 Konektivitas yang terbatas

Perangkat *mobile* memiliki *bandwith* rendah, beberapa dari mereka bahkan tidak te*RSA*mbung.

7 Masa hidup yang pendek

Perangkat-perangkat konsumen ini menyala dalam hitungan detik, bahkan kebanyakan dari mereka selalu menyala[7].

2.9 Penjelasan Umum SQLite

SQLite merupakan sebuah system manajemen basis data relasional yang bersifat ACID - *compliant* dan memiliki ukuran pustaka kode yang relatif kecil, ditulis dalam bahasa C.

SQLite merupakan proyek yang bersifat *public Domain* yang dikerjakan oleh D. Richard Hipp. Tidak seperti pada paradigma client-server umumnya, Inti SQLite bukanlah sebuah *system* yang mandiri yang berkomunikasi dengan sebuah program, melainkan sebagai bagian integral dari sebuah program secara keseluruhan. Sehingga protokol komunikasi utama yang digunakan adalah melalui pemanggilan API secara langsung melalui bahasa pemrograman. Mekanisme seperti ini tentunya membawa keuntungan karena dapat mereduksi *overhead*, *latency times*, dan secara keseluruhan lebih sederhana. Seluruh elemen basisdata (definisi data, tabel, indeks, dan data) disimpan sebagai sebuah *file*. Kesederhanaan dari sisi disain tersebut bisa diraih dengan cara mengunci keseluruhan *file* basis data pada saat sebuah transaksi dimulai.

SQLite merupakan proyek yang bersifat *public domain* yang dikerjakan oleh D. Richard Hipp. Tidak seperti pada paradigma client-server umumnya, Inti SQLite bukanlah sebuah *system* yang mandiri yang berkomunikasi dengan sebuah program, melainkan sebagai bagian integral dari sebuah program secara keseluruhan. Sehingga protokol komunikasi utama yang digunakan adalah melalui pemanggilan API secara langsung melalui bahasa pemrograman. Mekanisme seperti ini tentunya membawa keuntungan karena dapat mereduksi *overhead*, *latency times*, dan secara keseluruhan lebih sederhana. Seluruh elemen basisdata

(definisi data, tabel, indeks, dan data) disimpan sebagai sebuah *file*. Kesederhanaan dari sisi disain tersebut bisa diraih dengan cara mengunci keseluruhan *file* basis data pada saat sebuah transaksi dimulai[13].

2.10 Pengertian UML

UML singkatan dari *Unified Modeling Language* merupakan salah satu alat bantu yang dapat digunakan dalam bahasa pemograman yang berorientasi objek, saat ini UML akan mulai menjadi standar masa depan bagi industri pengembangan *system*/perangkat lunak yang berorientasi objek sebab pada dasarnya UML digunakan oleh banyak perusahaan raksasa seperti IBM, *Microsoft*, dan sebagainya[4].

2.10.1 Definisi UML

1 *Unified Modeling Language* merupakan metode pengembangan perangkat lunak (*system* informasi) dengan menggunakan metode grafis serta merupakan bahasa untuk visualisasi, spesifikasi, konstruksi serta dokumentasi.

2 *Unified Modeling Language (UML)* adalah bahasa yang telah menjadi standard untuk visualisasi, menetapkan, membangun dan mendokumentasikan arti suatu *system* perangkat lunak.

3 *Unified Modeling Language (UML)* dapat didefinisikan sebagai sebuah bahasa yang telah menjadi standar dalam industri untuk visualisasi, merancang dan mendokumentasikan *system* perangkat lunak.

4 *Unified Modeling Language (UML)* merupakan standard modeling language yang terdiri dari kumpulan-kumpulan *Diagram*, dikembangkan untuk membantu para pengembang *system* dan*software* agar bisa menyelesaikan tugas-tugas seperti :

   a. Spesifikasi
   b. Visualisasi
   c. Desain arsitektur
   d. Konstruksi

e. Simulasi dan testing

f. Dokumentasi

Berdasarkan beberapa pendapat yang dikemukakan diatas dapat ditarik kesimpulan bahwa "U*nified Modeling Language (UML)* adalah sebuah bahasa yang berdasarkan grafik atau gambar untuk menvisualisasikan, menspesifikasikan, membangun dan pendokumentasian dari sebuah *system* pengembangan perangkat lunak berbasis Objek (*OOP) (Object Oriented programming)"*[4].

2.11 Java

Java adalah sebuah teknologi yang diperkenalkan oleh *Sun Microsystems* pada pertengahan tahun 1990. Menurut definisi dari *Sun, Java* adalah nama untuk sekumpulan teknologi untuk membuat dan menjalankan perangkat lunak pada komputer *stand alone* ataupun pada lingkungan jaringan. *Java* berdiri di atas sebuah mesin *interpreter* yang diberi nama *Java Virtual Machine* (JVM). JVM inilah yang akan membaca *bytecode* dalam *file .class* dari suatu program sebagai representasi langsung program yang berisi bahasa mesin.

Oleh karena itu, bahasa *Java* disebut sebagai bahasa pemrograman yang *portable* karena dapat dijalankan pada berbagai *system* operasi, asalkan pada *system* operasi tersebut terdapat JVM. *Platform Java* terdiri dari kumpulan *library*, JVM, kelas-kelas *loader* yang dipaket dalam sebuah lingkungan rutin *Java*, dan sebuah *compiler*, *debuger*, dan perangkat lain yang dipaket dalam *Java Development Kit* (JDK)[1].

Java merupakan salah satu bahasa pemrograman yang menggunakan paradigma pemrograman berbasis objek. Paradigma ini yaitu menggunakan objek untuk membungkus atribut dan operasi yang mungkin pada objek tersebut. *Java* mempunyai kelebihan dan kekurangan yaitu:

1 Kelebihan Java

Beberapa kelebihan dari *java* yaitu sebagai berikut:

1. *Multyplatform*

   Kelebihan utama dari *java* ialah dapat dijalankan di beberapa *platform*/*system* operasi komputer, sesuai dengan prinsip tulis sekali, jalankan dimana saja. Kelebihan ini memungkinkan sebuah program berbasis *java* dikerjakan diatas *system* operasi *linux* tetapi dijalankan dengan baik di atas *Microsoft Windows*.

2. OOP (*Object Oriented Programming*)

   Java merupakan salah satu bahasan pemrograman berbasis objek secara murni. Semua tipe data diturunkan dari kelas dasar yang disebut objek. Hal ini sangat memudahkan pemrogram untuk men*desain*, membuat, mengembangkan, dan mengalokasi kesalahan sebuah program dengan basis *java* secara cepat, tepat, mudah dan terorganisir.

3. *Library* yang lengkap

   Java terkenal dengan kelengkapan *library*/perpustakaan (kumpulan program-program yang disertakan dalam pemrogram *java*) yang sangat memudahkan dalam *user* oleh para pemrogram untuk membangun aplikasinya. Kelengkapan perpustakaan ini ditambah dengan keberadaan komunitas *java* yang besar yang terus menerus membuat perpustakaan-perpustakaan baru untuk melingkupi seluruh kebutuhan pembangunan aplikasi.

4. Bergaya C++

   Java memiliki sintaks seperti bahasa pemrograman C++ sehingga menarik banyak pemrogram C++ untuk pindah ke *java*. Saat ini *user java* sangat banyak, sebagian besar adalah pemrogram C++ yang pindah ke *java*.

5. Pengumpulan sampah otomatis

   Java memiliki fasilitas pengaturan *user memory* sehingga para pemrogram tidak perlu melakukan pengaturan *memory* secara langsung (seperti halnya dalam bahasa C++ yang dipakai secara luas)[5].

2 Kekurangan Java

Beberapa kekurangan *Java* antara lain :

1. Mudah didekompilasi

   Dekompilasi adalah proses membalikkan dari kode jadi menjadi kode sumber. Ini memungkinkan karena kode jadi *java* merupakan *bytecode* yang menyimpan banyak atribut bahasa tingkat tinggi seperti nama-nama kelas, metode dan tipe data.

2. *User memory* yang banyak

   User memory untuk program berbasis *java* jauh lebih besar daripada bahasa tingkat tinggi generasi sebelumnya seperti C/C++ dan *Pascal* (lebih spesifik lagi *delphi* dan Objek *Pascal*)[5].

2.12 Android

Beberapa tahun belakangan ini, Android sedang naik daun, dan dapat dikatakan merupakan *system* operasi yang sedang berkembang pesat di dunia. Perkembangan Android tidak terlepas dari berkembangnya pasar *mobile phone* dan tablet yang sedang *booming* belakangan ini. Hal tersebutlah yang mengangkat pamor *system* operasi Android yang terkenal juga karena sifatnya *open source*, yang berarti berbagai developer perangkat *mobile phone* dan tablet dapat mengembangkan sistem operasi tersebut secara bebas[7].

2.12.1 Sejarah Android

Pada tahun 2003, Andy Rubin, Rich Miner, Nick Sears, dan Chris White bekerja sama membuat sistem operasi untuk perangkat *mobile*. Walaupun dimulai dengan tidak begitu menjanjikan , segalanya berubah dengan cepat ketika Andy Rubin – yang sudah terlebih dahulu memiliki sistem operasi yang dijalankan pada sebuah *mobile phone* yang dikenal dengan *Sidekick* – membawakan kuliah yang dihadiri oleh Larry Page dan Sergei Brin, pendiri Google. Kedua orang ini sangat tertarik dengan materi yang dibawakan oleh Rubin. Meskipun saat itu Google samasekali tidak membuat ponsel saat itu.

Pada Juli 2005, dengan kemampuan dana yang besar, tidak sulit bagi Google untuk mendapatkan perusahaan kecil myang didirikan Rubin. Di bawah kendali Google, perjalanan Android dimulai pada tahun 2007, Google mulai mendaftar paten di bidang telepon seluler. Langkah Google ini tentu mengundang banyak perhatian dan segera beredar kabar bahwa Google akan segera membuat ponsel untuk bersaing dengan ponsel-ponsel yang ada. Rumor tersebut ternyata keliru besar karena Google tidak membuat ponsel waktu itu, tetapi mengembangkan sistem operasi yang akan digunakan untuk ponsel.

Ponsel pertama yang menggunakan sistem operasi dari Google justru datang dari HTC dengan seri HTC Dream atau T-Mobile G1 yang diluncurkan pada 2008. Google sendiri baru mengeluarkan ponselnya sendiri yang dinamakan Nexus One pada tahun 2010. Ponsel tersebut tidak dibuat sendiri oleh Google, melainkan merupakan hasil kerja sama dengan HTC.

Berbeda dengan sistem operasi lain, Android berkembang sangat cepat dan secara konsisten mengeluarkan versi-versi yang terus diperbarui dalam waktu yang cukup singkat. Nama–nama yang digunakan untuk versi Android-pun cukup unik karena menggunakan nama makanan dan huruf awalnya disusun berdasarkan huruf abjad sehingga mudah untuk mengetahui versi mana yang lebih baru. (*Cupcake, Donut, Eclair, Froyo, Gingerbread, Honeycomb, Ice Cream Sandwich,* dan seterusnya)[7].

2.12.2 Lisensi pada Android

Sistem operasi Android adalah sistem operasi yang *open source*, tetapi pada kenyataannya Google juga menerapkan lisensi terhadap pembuat *hardware* yang menggunakan Android, yang tentunya sedikit banyak akan mempengaruhi terhadap layanan yang akan didapatkan oleh *user*.

Google menerapkan dua jenis lisensi, yaitu GMS (*Google Mobile Services*) dan OHD (*Open Handset Develpment*). GMS didukung secapa penuh oleh Google di mana user akan mendapat sistem operasi Android yang terhubung dengan layanan Google *Account*. Terhubung dengan Google *Account*, akan membuat perangkat yang digunakan oleh user mendapatkan berbagai layanan

seperti Android *Market*, Gmail, Google *Contact*, Google *Calendar*, dan Google *Maps*. Layanan–layanan Google tersebut adalah daya tarik utama sistem operasi Android. Lisensi GMS, membutuhkan biaya yang harus dibayarkan oleh vendor *hardware* kepada Google.

Berbeda dengan lisensi GMS, lisensi OHD, vendor bisa menggunakan system operasi Android secara gratis tanpa harus membayar kepada Google. Karena alasan inilah, banyak vendor hardware menggunakan lisensi ini. Kekurangan utama dari jenis lisensi OHD adalah tidak adanya dukungan yang diberikan oleh Google, sehingga user tidak dapat menikmati layanan seperti Android Market, Google Maps, dll. Walaupun tidak mendapat dukungan dari Google, *mobile phone* atau perangkat yang mengunakan lisensi OHD tetap bisa menggunakan aplikasi-aplikasi yang dibangun untuk Android. Hal tersebut dapat terjadi karena sistem Android sangat bebas, di mana user bisa mendapatkan aplikasi dari tempat lain secara sah, tanpa melalui Android *Market*[7].